International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I8P110

Volume 73 Issue 8, 117-128, August 2025
© 2025 Seventh Sense Research Group®

Review Article

A Comparative Analysis of Deep Belief Networks: Baseline and Optimized Architectures for Fine-Tuning

Raji. N[1], S. Manohar[2]

[1,2]*Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India .*

[1]*Corresponding Author : rn5525@srmist.edu.in*

Received: 12 February 2025 Revised: 10 July 2025 Accepted: 21 July 2025 Published: 30 August 2025

Abstract - *This study discusses the comprehensive analysis of Deep Belief Networks, which mainly focuses on comparing unoptimized and optimized architectures and comparing the performance on various datasets. Optimized DBN addresses some of the limitations in this regard by using advanced techniques like adaptive learning rates, regularization strategies, sparsity, and pruning to prevent problems like vanishing gradients, computational inefficiency, and sensitivity to hyperparameters. Empirical results show that the optimized DBN has considerable performance improvements, with an accuracy of over 98% against 92% for the unoptimized counterpart. All the metrics, including precision, recall, F1-score, AUC, and log-loss, also showed considerable gains. Evaluations across multiple datasets, including medical imaging and agricultural data, confirm the robustness and generalization of the optimized DBN. Moreover, K-fold validation emphasizes the stability of these improvements and demonstrates the optimized DBN's strength in classification. This work further emphasizes the significance of optimization in improving the performance of DBNs and provides a framework for developing efficient and scalable deep learning models.*

Keywords - *Deep Belief Networks (DBNs), Optimized architectures, Hyperparameter sensitivity, Regularization strategies, K-fold validation.*

1. Introduction

Deep learning has emerged as an innovative approach in the development of artificial intelligence to revamp supervised classification tasks for every domain. By the employment of hierarchical architectures, the deep learning models could learn feature representation automatically from raw data that is much more complex. Deep learning has become indispensable in several applications like image, Speech recognition, medical diagnoses, and recognition that require meaningful patterns from the extracted high-dimensional data to accurately [1, 2]. Among all these deep architectures, Deep Belief Networks (DBNs) are one of the architectures that take a significant place as they possess the property to combine both unsupervised and supervised learning. Stacked by RBMs, or restricted Boltzmann machines, DBNs are proficient at gaining knowledge of the hierarchical representations of features via unsupervised pre-training followed by fine-tuning with labeled data [3-5]. As such, dual-phase training makes DBNs particularly successful in situations when labeled data is rare or hard to come by. In addition, their architectural structure is beneficial in capturing complex data patterns and relationships, rendering them as a powerful model for difficult classification tasks. Although these have their strengths, several issues limit the efficiency and Scalability of DBNs. They suffer from computational inefficiency, sensitivity to hyperparameter settings, and difficulties in fine-tuning across deep layers. Different optimization techniques have been suggested to overcome such limitations, including adaptive learning rates, regularization methods, and pruning strategies. These techniques are applied to enhance fine-tuning, improve generalization, reduce Overfitting, and lower computation overheads, enabling robustness and suitability in real-world applications [6-8]. The current research aims to provide an overall comparative analysis between the baseline DBNs and their optimized versions. While discussing the progress on architectures, training methodologies, and impacts of optimization techniques, it identifies the improvements in computational complexity, learning efficiency, and classification performance. The conclusions derived from this comparison would guide the evolution of deeper, more effective, and more data-driven knowledge models for supervised classification tasks.

2. Background

2.1. Transition from Shallow to Deep Architectures

2.1.1. Shallow Networks

At the ANN at the nodal level, the simplistic McCulloch-Pitts neural model was developed in 1943. consisted of a deterministic binary activation function and a simple summing unit [9]. The complexity increased with each repetition by the successors. Gaussian, sigmoid, and linear functions were used at the activation function level. The complex domain was now included in the outputs previously limited to real values. In order to simulate ionic exchanges, stochastic neurons and spiking neurons replaced deterministic models. In order to create more complex learning models, all of these additions were made. Single-layered topologies, such as Aizerman's kernel perceptron (1964), the ADALINE network by Widrow and Hoff (1960) [10], and Rosenblatt's perceptron (1957) [11], were the first to be implemented at the network level. The XOR issue is a straightforward challenge of non-linear binary classification that these architectures struggled to learn and performed poorly. As a result, increasingly sophisticated networks were introduced, beginning with the Kohonen networks, sometimes known as self-organizing maps (SOM), [12], (1986), Perceptron with many layers [13] Hopfield networks that are self-recurrent (Rumelhart, 1986) [14] (1986), Adaptive Resonance Theory (ART) networks (1980s) [15], and several others that are regarded as architectures that are shallow because of their many hidden layers.

Higher levels of intelligence were promised by successive iterations, which gradually improved on the shortcomings of their predecessors. This claim was made partially possible by the hardware's increased computing power and the creation of quicker and more effective algorithms for learning and training. Both (back propagation) supervised and unsupervised algorithms that use feed-forward learning mechanics were developed concurrently and improved performance over a variety of particular tasks. However, the combined impact of the innovation that focused on every facet among these superficial networks was insufficient to catch a person's intellect fully, and the advancement of deeper networks was slowed down by high computational requirements. The following illustrates the shortcomings of shallow networks.

Inability to Capture Complex Patterns

Shallow networks that are fewer in layers cannot really depict complex, hierarchical data representations, such as those involved in image recognition, natural language processing, or speech recognition. They are usually able to learn only linear or very simple non-linear functions that may not be enough for datasets with high dimensionality and a more structured nature.

Curse of Dimensionality

In order to achieve acceptable performance, the more input features there are, the more neurons shallow networks require, and they consume much more data; consequently, the computational and memory demands grow exponentially. As such, the approach results in inefficiency when working with high-dimensional or large-scale data.

Poor Generalization on Non-linear Data

Shallow nets are not able to distinguish overlapping features from non-linear datasets successfully and, therefore, may overfit or generalize too poorly.

Limited Hierarchical Feature Extraction

The deep nets can simultaneously learn multiple abstract layers, which are considered indispensable for interpreting complex data representations. Since the feature hierarchies (edges-to-shapes-to-objects in images) are pertinent in many applications, shallower architectures are not quite fit for these applications.

Lack of Robustness to Variations in Data

Shallow networks are too sensitive to noise and variations within the input data, as their depth is not sufficient to learn invariant features and, hence, may incur instability in performance across diverse datasets or environments.

Vanishing Gradient Problem

While not as drastic as with deeper architectures, even shallow networks with a few hundred layers will suffer from an inability to propagate gradients during training under certain activation functions effectively.

Limited Scalability

Such shallow networks cannot easily accommodate big, complex datasets or tasks without greatly increasing the number of neurons and connections, which leads to inefficiencies and higher chances of Overfitting.

Inflexibility for Transfer Learning

Unlike deep architectures, shallow networks are not amenable to transfer learning since they do not have layered representations that allow them to adapt to new tasks or domains. These limitations explain why deep architectures, particularly DBNs, are relevant because they solve the constraints by introducing more layers so that complex patterns and their hierarchical data relationships can be modeled.

2.1.2. Deep Architectures

As processing power improved and more effective training algorithms were developed, training deep architectures became possible, leading to a resurgence in ANN research in the early 2000s [16]. The training process for Boltzmann machines was made simpler by the greedy training algorithm of Hinton et al., while deep stacking networks reduced the computing load by breaking down training into the deep network's building blocks. Additionally, training deeper recurrent neural networks was made possible by the architecture of Schmidhuber's extended short-term memory [17]. Although the biological properties of the brain beyond the neuron are not borrowed by these architectures, the force gained in the field of computational neuroscience is helping the connectionist community embrace neural network topologies in deep designs that are more in line with neuroscientific hypotheses about the structure of the human brain. A physiologically sound method for learning and producing invariant representations of sensory patterns was absent from earlier ANNs.

2.2. Introduction of DBNs as a Solution for Hierarchical Feature Learning

Unlike ML approaches, DNNs are more complex versions of shallow ANN designs, which consist of a mathematically simplified representation of a biological neuron. They do not, however, aim to replicate the human brain precisely. DNNs are based on noncognition, an image processing model with biological inspiration that aims to achieve powerful AI models by using hierarchical knowledge abstraction. Shallower layers pick up low-level statistical patterns as data moves through the network, and Deeper layers pick up more complex and abstract representations by building on these properties. Due to their inability to make clear logical deductions, DNNs need more development to include abstract knowledge in a way that is human-like. ANNs were initially trained via an algorithm called backpropagation, which propagates the output error backwards through the network to change the network weights. As network depth increases, the inaccuracy that spreads disappears to zero, stopping the updating early-layer weights and drastically lowering network performance. Other DNN training strategies were thus examined. To train recurrent neural networks, Schmidhuber [18] (1992) suggested pre-training layers in an unsupervised manner and then using backpropagation to fine-tune the network weights. A greedy training algorithm specifically for DBN was suggested, which further accelerated the momentum. Although the DBN is a kind of deep ANN, its architectural features prevent backpropagation from generating a model that works effectively on training and testing data. This phenomenon has been ascribed to “explaining away.” To put it another way, sometimes referred to as selection bias or Berkson’s paradox, invalidates the widely accepted notion of layer independence and hence complicates the inference process [19]. Since it is impossible for both concealed nodes to fire at the same time due to their incredibly low probability, they become anti-correlated. In order to address this problem, a training technique was developed based on the finding that DBN may be reduced to inter-layer neuron connections in a two-layer network that are exclusively and successively layered restricted Boltzmann machines (RBM). Automatic speech recognition, image processing, natural language processing, feature extraction and reduction, and many more problems were resolved with DBN. This innovative method reignited interest in these deep architectures.

2.3. Challenges in DBNs

Although DBNs have many benefits in terms of hierarchical feature representation and learning, they are not without drawbacks. Key concerns consist of:

2.3.1. Vanishing Gradients

One of the primary issues in training deep architectures like DBNs is the vanishing gradient problem. Backpropagation gradients decrease during loss function computation as they move through the layers, especially in deep networks. This results in almost negligible updates to the weights in the earlier layers and, therefore, does not allow the model to learn effectively from data.

2.3.2. Overfitting

DBNs, especially if the training dataset is small, have a tendency to overfit. Overfitting means the model learned how to commit the training data to memory instead of generalizing it. Overfitting reduces the performance of the model on unknown data and limits its practical usability.

2.3.3. Computational Cost

DBNs are computationally expensive, particularly during their pre-training and fine-tuning. The computational requirement for multiple iterations across layers and several adjustments across parameters is pretty demanding for DBNs. Therefore, their efficiency is limited when a large dataset needs to be processed or when it needs real-time processing.

2.3.4. Sensitivity to Hyperparameters

Hyperparameter tuning-very strong DBNs are sensitive, but for the fine tuning to go correctly, it depends on some careful setup regarding these- learning rate, and many more numbers of units within a layer and layer, etc. Sub-optimal choices often lead to a much worse training set.

2.3.5. Scalability

With the increasing size of the network, the memory and computation requirements rise exponentially. Thus, scaling DBNs to extremely deep architectures or very large datasets proves difficult. These are mitigated through optimization techniques that have been developed, such as adaptive learning rates, regularization methods, such as Dropout, and pruning strategies that reduce the complexity of models while maintaining their performance.

3. DBN- Overview

3.1. Description of DBN as a Stack of RBMs

Professor Hinton created DBN, a deep architecture based on RBM. DBN can partially address the issues of lengthy training times and challenging depth model optimization since RBM, a component of the depth model, exhibits greedy training layer by layer [20].

3.1.1. Architecture of Restricted Boltzmann Machine

Originally created by Hinton and Sejnowski, Boltzmann machines (BM) showed a remarkable capacity for unsupervised learning as a generalized version of “connectionism.” They did, however, have some drawbacks, like postponed training. Smolensky developed an RBM model to address the limitations of BM [21]. The visual layer, a visible component of RBM, is where data, represented by v, is received. The hidden unit, sometimes referred to as the hidden layer, is the other component. It is denoted by h and is employed to extract features from data. Though there is no link within the layer, both layers are connected. Figure 1 displays the RBM model's structure. If the value (v, h) corresponding to the input dataset is assumed to follow the mathematical Bernoulli distribution for both the visible and hidden layer units, then the RBM energy function can be expressed as follows:

$$E(v,h;\theta) = -\sum_{i=1}^{I} a_i v_i - \sum_{j=1}^{J} b_j h_j - \sum_{i=1}^{I}\sum_{j=1}^{J} v_i w_{ij} h_j \quad \theta = \{w_{ij}, a_i, b_i\} \tag{1}$$

Where θ are the RBM model's parameters, sometimes referred to as the energy function. Consequently, the following definition of the combined probability distribution of every layer that is visible and hidden can be made:

$$\begin{Bmatrix} P(v,h;\theta) = \frac{1}{Z(\theta)} \exp\left(-E(v,h;\theta)\right) \\ Z(\theta) = \sum_{v,h} \exp\left(-E(v,h;\theta)\right) \end{Bmatrix} \tag{2}$$

The allocation function, normalized factor Z(霓), is derived by adding up all nodes' energy indices.

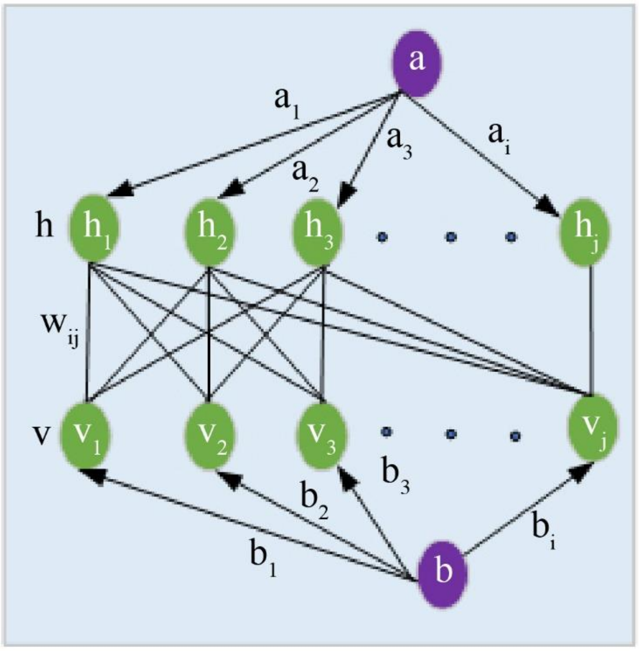

Fig. 1 Structure of RBM model

The distribution function's definition makes clear how difficult it is to compute directly. The RBM layer has no relationship, hence all items in the visible and buried layers are independent of conditions. Therefore, Z can be found by using the concepts of the probability distribution. The RBM model's visual layer vs. likelihood function, or edge distribution expression, is as follows:

$$p(v;\theta) = \sum_h p(v,h;\theta) = \frac{1}{Z(\theta)} \sum_h \exp\left(-E(v,h;\theta)\right)$$

$$= \frac{1}{Z(\theta)} \sum_h \exp\,(v^T wh + b^T v + a^T h) \tag{3}$$

The conditional likelihood that is visible vector v and hidden unit h is:

$$p(h|v,\theta) = \frac{p(v,h;\theta)}{p(v;\theta)}$$

$$= \frac{\frac{1}{z(\theta)} \exp\,(b^T v) \prod_j exp\left(a_j h_j + \sum_{i=1}^{I} w_{ij} v_i h_j\right)}{\frac{1}{z(\theta)} \exp\,(b^T v) \prod_j \sum hexp\left(a_j h_j + \sum_{i=1}^{I} w_{ij} v_i h_j\right)} \tag{4}$$

The visible unit v and the hidden vector h have comparable conditional probabilities:

$$p(v|h;\theta) = \frac{p(v,h;\theta)}{p(h;\theta)} = \prod_i p(v_i|h) \tag{5}$$

Since there is no inter-layer link, the order multiplication serves as the conditional probability and may be written as:

$$\begin{Bmatrix} p(h_j = (1|v) = \sigma\left(a_j + \sum_i v_i w_{ij}\right) \\ p(v_i = (1|h) = \sigma\left(b_i + \sum_j h_i w_{ij}\right) \end{Bmatrix} \tag{6}$$

Where σ is the sigmoid function; the range is (0, 1); the definition domain is $(-\infty, +\infty)$. Upon receiving the training data, RBM first applies Eq. (13), which translates characteristics of the input data from the hidden layer (h) to the visible layer (v). The output data is subsequently rebuilt from the hidden layer to the visible layer using Equation (14) as a guide. Until the computation criteria are reached, the RBM model's parameter {,*-ij*.,,*a-i*., *b-j*. } is continuously adjusted based on the error.

The difference between the original and currently rebuilt data is now calculated. By reducing the error between the input and reconstructed data, RBM training seeks to maximize the likelihood function. Since it is challenging to solve directly, stochastic gradient descent—which uses the following formula—is how we learn:

$$\theta = \theta + \varepsilon \frac{\partial In(\theta)}{\partial \theta} \tag{7}$$

Although it is challenging to compute, the parameters in θ are typically calculated in the form of block Gibbs sampling. Consequently, the contrastive divergence (CD) technique was presented by Hinton and is frequently used due to its quick learning speed.

3.2. Network Architecture

RBM is arranged in series to form the DBN model. RBM is trained in order to accomplish the pre-training of DBN during training, layer by layer, from low to high. The structure of the DBN model is shown in Fiure. 2. The visible layer of the first RBM receives the input data. Its hidden layer is concurrently the visible layer of the RBM that is connected to it, and its output is the input of the succeeding RBM. In the end, this leads to the creation of the DBN, or deep network architecture with multiple hidden layers.

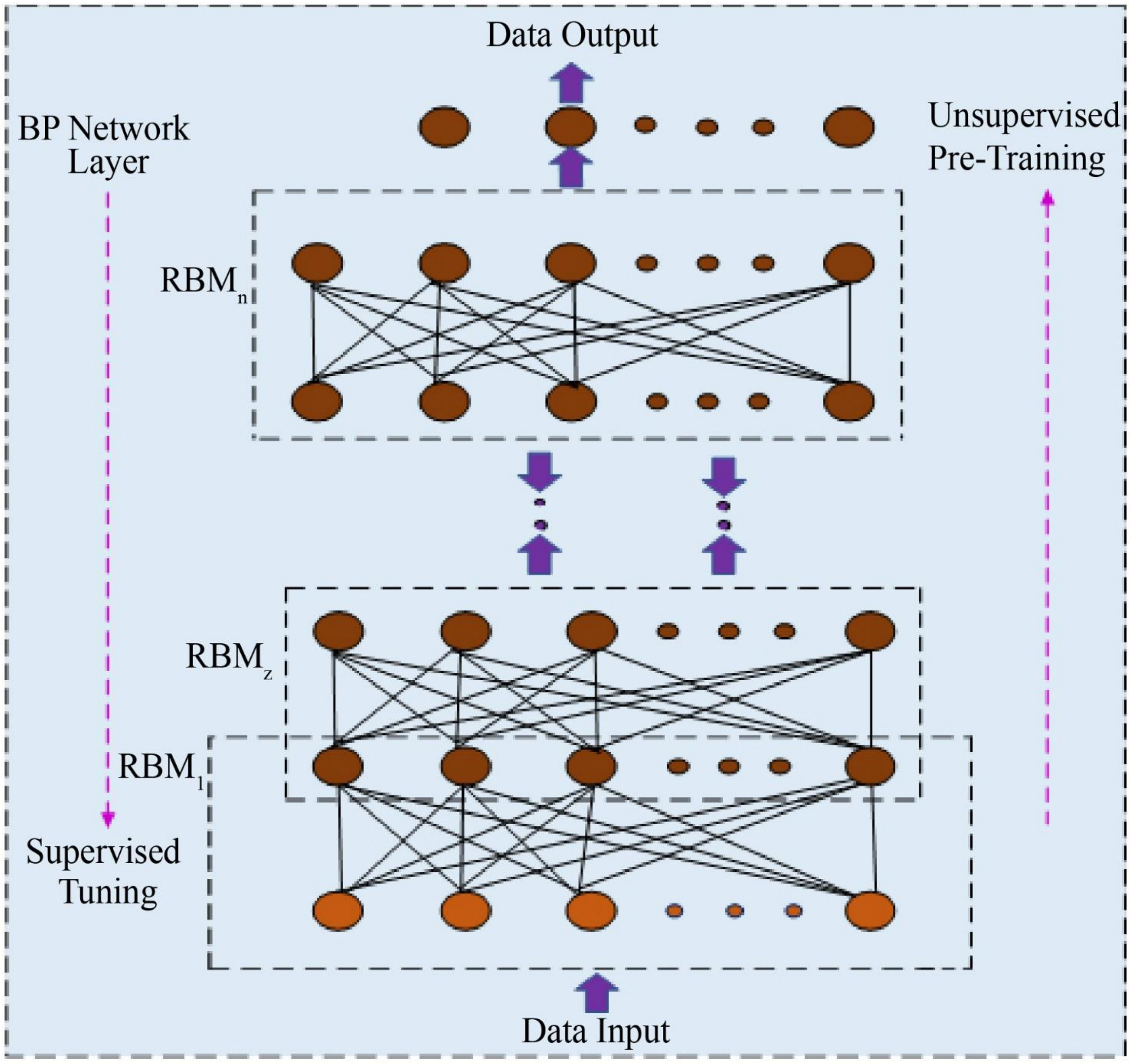

Fig. 2 Structure of DBM model

3.2.1. Layer-wise Unsupervised Pre-Training and BP Supervised Training

Unsupervised Pre-Training Phase

When the predetermined number of iterations is attained, connecting to the first DBN network which is only locally optimal. The bias parameters and weights of each layer's hidden and visible layers of the RBM network are trained independently and unsupervisedly. In other words, each RBM layer only reaches its maximum.

Initialization of Parameters

Give the training sample first, followed by the model parameters (learning rate, number of visible and hidden layer units, etc.). Set to its initial value, the RBM model parameter { }= {,w-ij.,,a-i., b-j. }. This relates to the connection weights between layers as well as the bias of each layer. The following calculation procedure is typically used to initialize parameter θ using the random minimum value:

$$\begin{Bmatrix} w = 0.1 \times randn(n,m) \\ a = 0.1 \times randn(1,n) \\ b = 0.1 \times randn(1,m) \end{Bmatrix} \tag{8}$$

The model parameters (number of visible and hidden layer units, learning rate, etc.) and the training samples should be provided initially. Set the RBM model parameter to its starting value. Each layer's bias and the connection weights between layers are correlated with this. The standard method for initializing parameter θ with the random minimum value is the following calculation process.

4. Training

Determine After adding parameter setup values to the network input, the training sample "Train.mat" loop uses connection weights, $w\text{-}ij$, and offset ai and bj. The contrast scatter technique, used in RBM training, generates a training sample from the DBN's initial visible layer, (0). network after "Train.mat" is entered. Applying Eq. (9) to the subsequent hidden unit linked, the probability value between 0 and 1 is determined, followed by the observed probability value. Next, we compute the observed unit's probability value using Equation (9) to get $v(1)$. Here are the precise steps:

Step 1: Use Equation (9) to predict the likelihood that a hidden layer unit h_j Will be triggered, then use the following Equation to ascertain the neuron's condition in that layer:

$$h_j = \begin{Bmatrix} 1, if \quad r_j < P(h_i = 1|v), \quad r_j \in [0,1]_{(j=1,2,\ldots\ldots,J)} \\ 1, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad otherwise \end{Bmatrix} \tag{9}$$

Step 2: In order to recon Figure. The visible layer, step two, involves calculating the likelihood that the visible layer unit vi will be activated using Equation (14) and determining the neuron's condition within the layer using Equation (10).

$$v_i = \begin{Bmatrix} 1, if \quad r_j < P(v_i = 1|h), \quad r_j \in [0,1]_{, \ (i=1,2,\ldots\ldots,J)} \\ 1, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad otherwise \end{Bmatrix} \tag{10}$$

Afterwards, by comparing the Contrastive Divergence (CD), one can get the approximate value of Equation (11) procedure:

$$\begin{Bmatrix} \Delta w_{ij} = \frac{\partial ln\ L(\theta)}{\partial\ w_{ij}} \approx P(h_j = 1|v^{(0)})v_i^{(0)} - P(h_j = 1|v^{(1)})v_i^{(1)} \\ \Delta a_i = \frac{\partial ln\ L(\theta)}{\partial\ a_i} \approx v_i^{(0)} - v_i^{(1)} \\ \Delta b_j = \frac{\partial ln\ L(\theta)}{\partial\ b_j} \approx P(h_j = 1|v^{(0)}) - P(h_j = 1|v^{(1)}) \end{Bmatrix} \tag{11}$$

The CD technique was used to reconstruct the data, and the network parameters were then modified using the following Equation.

$$\begin{Bmatrix} w_{ij}(k+1) \leftarrow w_{ij}(k) + \varepsilon\Delta w_{ij} \\ a_i(k+1) \leftarrow a_i(k) + \varepsilon\Delta a_i \\ b_j(k+1) \leftarrow b_j(k) + \varepsilon\Delta b_j \end{Bmatrix} \tag{12}$$

To generate the results, repeat the previous procedures until the specified training sessions have been completed.

4.1. BP Supervised Training

During the reverse fine-tuning phase, the output feature vector from the RBM is used as the input feature vector by the top-level processor (BP). Then, depending on the computation of the BP network's output error e and error energy total E, the weights $w\text{-}ij$. And bias thresholds ai and $b\text{-}j$. Are modified. The formulas are as follows:

$$\begin{Bmatrix} w'_{ij} = w_{ij} + \eta \frac{\partial E}{\partial w_{ij}} \\ a'_i = a_i + \eta \frac{\partial E}{\partial a_i} \\ b'_j = b_j + \eta \frac{\partial E}{\partial b_j} \end{Bmatrix} \tag{13}$$

Following the completion of the aforementioned tasks, DBN begins predicting the "Test.mat" test set and produces the results of the prediction.

4.1.1. Application of DBN

The softmax layer uses the Gradient Descent in Evolution (EGD) technique to categorize the extracted features [22]. In contrast, the stacking restricted Boltzmann machine uses the contrastive divergence method to extract features. When compared to the Gradient Descent process, the EGD acceleration rate is impressive. DBNs have been effectively used in numerous fields, including:

- Image recognition is the process of identifying and classifying objects by extracting features from raw pixel data.
- Speech Processing: Enhancing automatic speech recognition systems by learning robust audio feature representations.
- Natural Language Processing: Improving text understanding and semantic analysis by capturing contextual relationships.
- Healthcare: Assisting in tasks such as disease diagnosis by processing and analyzing medical images or sensor data.

4.1.2. Limitations in the Standard DBN Approach

According to DBN, its execution requires significant space and high energy. The enormous demand for Random Number Generators (RNGs) reduces deep belief networks' energy efficiency.

Additionally, the backpropagation neural network with multiple hidden layers in DBN requires more time to learn. Because the posterior is not factorizable in every training scenario, greedy learning is thus ineffective in the directed module. In a sigmoid belief network, learning in layers is difficult due to the general placement of higher variables at points before the first hidden layer. These algorithms' primary obstacles are the lack of training data for intrusion detection, data imbalances in ad hoc applications, The inability to comprehend data in a variety of biological applications, the increase in uncertainty in the healthcare sector, Model compression in healthcare, catastrophic forgetfulness in biological and other applications, Data under specification in safe routing, Exploding Gradient and the Vanishing Gradient problem in energy-efficient network development, and over-fitting during misbehavior identification in medical applications of a mobile network.

4.2. The Role of Fine-Tuning

One of the most important complicating factors that arises in DBN training is the potential degradation problem. The problem reveals itself in practice during repetitions of training iterations as a log-likelihood decay of the model. Identifying this deterioration is necessary because it disrupts the DBNs' general performance as well as effectiveness in numerous applications [23]. The term "likelihood degradation" describes how the model's log-likelihood of identified data goes down with its training. This decline may indicate that the model loses its ability to accurately describe the statistical properties of the training set of data. Likelihood degradation causes reduced model accuracy, inferior generalization to new data, and a higher tendency for Overfitting. These effects compromise the dependability and practicality of DBNs in practical settings.

Conventional training methods like CD and its variants may induce uncertainty and weaken the likelihood in the case of long training periods. The reduction in likelihood can be made worse by poorly choosing hyperparameters like learning rates and regularization strengths. Degradation is more likely when DBNs become harder to train, as it becomes increasingly difficult to maintain training consistency. When a model is used to predict new, unvalidated data, its probability can decrease, and accuracy and trustworthiness may diminish. Increased tendencies of Overfitting, which the model performs well in training data but fails to generalize into test data, are generally correlated with a higher degradation in likelihood. Probability may drop, which means that the training can be very lengthy to converge, or it can demand more computation power to attain good levels of performance. However, even with their success, these methods can have pitfalls like log-likelihood deterioration in training that negatively impact the overall learning performance of DBNs. The establishment of adaptive practices for the reduction of likelihood deterioration and a better understanding of the methods that cause it must be developed to overcome such challenges. One interesting strategy is adaptive strategies for modifying and optimizing parameters, where regularization strength can be modified dynamically, thresholds can be updated, and learning rates during training can be adapted. Further quickening training and overall performance is likely with DBN efficacy optimized starting functions.

4.3. Importance of Supervised Fine-tuning for Achieving Optimal Performance

Supervised fine-tuning is the final critical stage in DBN training. The process fills the gap between unsupervised pre-training and task-specific optimization. It refines the representation, which was learned through unsupervised pre-training about hierarchical representations from raw data, to align it with labeled data and to boost performance on specific tasks. This phase is the final stage in achieving the highest classification accuracy and generalization. One of the main reasons that supervised fine-tuning is necessary is that it overcomes the disadvantages of unsupervised pre-training. Parameters trained under pre-training are local optimizers for feature extraction, but they are often not aligned with the required task. Supervised fine-tuning adjusts the parameters globally by minimizing the cross-entropy loss functions particular to the task, which means that the network will be outputting the values as consistent with the true ground truth labels.

Also, the supervised fine-tuning avoids overfitting and degradation of likelihood that may happen during the pre-training phase. It stabilizes the training procedure by using weight decay and dropout regularization methods, prevents overfitting to certain features, and improves generalization ability for unseen data. This is highly important in deep architectures since the more complex and deep the network is, the higher the risk of Overfitting. The fine-tuning process also allows for the incorporation of sophisticated optimization techniques. For example, adaptive learning rates, momentum-based gradient descent, and dynamic weight decay factors can be used during fine-tuning to speed up convergence and increase accuracy. These techniques improve the network's ability to effectively explore the parameter space, avoid suboptimal solutions, and achieve faster and more robust convergence.

5. Optimized DBN

5.1. Enhanced Training Techniques

This is the degradation of the likelihood during learning of the DBN. Therefore, an active area in preventing divergence during learning is the choice of the weight decay parameters. In this work, an adaptive weight decay factor with SaPO [24], which will be used for dynamic changes of regularization strength, will be developed to modify the learning rate dynamically during the adaptation. The adaptive weight update threshold will be introduced along with its training via adaptive updates of the learning rate of the DBN. Further, the application of the weighted activation function could enhance the training performance while reducing the time of computation.

The improved training methods proposed by the optimized DBN remove log-likelihood degradation, prevent Overfitting, and prevent slower convergence. Adaptive weights, through the Piecewise Chaos strategy, dynamically govern a decay factor to balance cross entropy so that the strength of regularization is effectively enhanced, avoiding overfitting and achieving optimal performance. Furthermore, an adaptive weight-update threshold dynamically changes the training rates to speed up training and avoid stagnating at a point or becoming divergent. The efficiency can be enhanced further by incorporating a weighted activation function that will assign the importance of activation to eliminate redundant computations and reduce the time required during training. All these methods are adopted during the phases of DBN training: first is the supervised fine-tuning phase after unsupervised pre-training using RBMs, in which fine-tuning will refine parameters. Together, these enhance generalization, reduce computational overhead, and considerably improve accuracy and efficiency for complex tasks involving machine learning.

5.2. Sparsity and Pruning

Sparsity and pruning are extremely important techniques in the optimization of DBNs towards the simplification of network complexity and, hence, in terms of computational efficiency. Sparsity is the ability to restrict the number of active neurons or weights so that only the most influential connections support learning and inference in the network. This actually decreases the usage of memory and decreases the computational cost as the model becomes quicker and more efficient, or even better at accuracy. It simply means pruning the systematic elimination of unnecessary or redundant weights and neurons within the DBN. This will make the network more streamlined because it only deals with the pathways that are supposed to carry data propagation and learning, thus making it a much smaller model and reducing the general computation needed. When combined, sparsity and pruning help create a more efficient DBN since they reduce the number of non-zero weights and optimize the resource utilization. It impacts memory use and processing power. This will allow real-time applications to scale up well and adapt better. The optimized model of fewer yet very effective connections performs well, but is also a lot more resource-efficient, a great requirement for deploying DBNs in situations with low computational capability.

5.3. Advanced Fine-Tuning Strategies

Advanced Fine-Tuning Techniques for fine-tuned DBNs focus on tuning pre-trained model parameters to maximize accuracy and generalization, and remove the inefficiencies of training. Fine-tuning is very much essential after the pre-training stage of unsupervised RBMs for optimizing the performance of the network using supervised learning techniques. This phase consists of the following advanced techniques:

5.3.1. Fine-tuning Phase

Only one RBM can obtain optimal parameters after the initial DBN model is generated by unsupervised RBM training. Cross-entropy is employed as the loss function, and an inverse BP network layer is added to the DBN tail layer in order to improve the parameter matrix of the network stacked by RBM. It fine-tunes the DBN network's parameter matrix that has been obtained at the pre-training stage in top-to-bottom supervision from error in label data and output, so that the DBN model's output could be maximized toward the original input.

Inverse Backpropagation Layer Addition

Adding an inverse backpropagation layer to the end of the DBN improves the optimization of parameters. This layer oversees the update of weights and biases in the entire stacked RBM layers, such that the learned features are aligned correctly with the target output.

Cross-Entropy Loss Function

In fine-tuning, cross-entropy serves as the main loss function. It monitors the discrepancy between the ground truth labels and the expected output, which is a good mechanism in supervised learning to ensure minimization of classification errors from the model.

Hybrid or Iterative Training Algorithms

Hybrid approaches like Momentum-based Gradient Descent (MGD) are used to fine-tune the parameters during training iteratively. The momentum term helps the algorithm avoid local minima and, thus, helps find a better learning trajectory.

Parameter Tuning

Pre-trained weights and bias, during training, parameters learned from them are finely tuned from the top down by systematic optimization within the DBN model network, while guaranteeing the top abstraction quality of this predictive accuracy model.

Dynamic Regularization and Learning Rates

Fine-tuning uses dynamic updates in learning rates and regularization strengths to avoid Overfitting. It reduces computational cost while speeding up convergence for the supervised training.

5.4. Benefits of Optimization

The optimization of DBN offers many significant benefits and is more efficient and effective for complex machine learning tasks. One such key advantage is the increase in accuracy and generalization, where using adaptive weight decay, adjustments of the learning rate, and advanced fine-tuning strategies reduce Overfitting and improve its ability to perform well on unseen data. Another benefit is that it reduces the computational overhead. Techniques such as sparsity, pruning, and weighted activation functions reduce the count of active parameters and irrelevant computations, which leads to more rapid training and inference time. Furthermore, optimization has improved the convergence speed of the model, with adaptive strategies that dynamically modify the learning parameters to avoid stagnation or divergence during the training process.

6. Theoretical Computational Complexity

The computational complexity of the Deep Belief Network and its optimized variant can be theoretically considered in terms of memory, processing requirements, and how optimization techniques introduce efficiency in their use.

6.1. DBN Complexity

- Non-Zero Weights and Memory Usage: The number of non-zero weights is an important factor determining a

DBN's complexity. A traditional DBN is rather heavy on memory because it uses weights, biases, and activation outputs across layers.

- Role of the Activation Functions: The activation function contributes to a computational expense, particularly for non-linear functions like the sigmoid or ReLU. Their complexity is proportional to that of neurons and layers in the network.
- Pre-Training Phase: The amount of visible and hidden units in each layer determines the pre-training phase cost of RBMs and the sampling iterations required to estimate the gradients.

6.2. Optimized DBN Complexity

6.2.1. Impact of Sparsity and Pruning

Sparsity reduces the number of active (non-zero) connections, which lowers both memory requirements and computation. Pruning of redundant weights and neurons eliminates, which reduces the size of the effective network and decreases matrix multiplication during forward and backwards passes.

6.2.2. Dynamic Regularization and Learning Rates

Adaptive weight decay reduces Overfitting and avoids unnecessary weight updates, thus avoiding redundant computations.

Dynamically adjusting the learning rate minimizes the number of epochs to convergence, meaning lower overall training time.

- Weighted Activation Functions: The focus on significant activations minimizes the computational cost per neuron, especially in dense layers.
- Fine-Tuning: Adding an inverse backpropagation layer during fine-tuning slightly increases complexity but allows for better optimization of the parameters, resulting in fewer epochs overall.

6.3. Comparison

The computational costs of basic DBN and optimized DBN are very different because of the advanced techniques applied in the latter. Basic DBNs, usually involving dense connections over 3–5 layers, consume large amounts of memory and computation to store and process all weights and biases. The training time for basic DBNs is relatively long (e.g., ~900 seconds) because of fixed learning rates and manual tuning of parameters, which results in slower convergence. In contrast, optimized DBNs can be made sparser, more prone to pruning, adaptive dynamic learning rate, and adaptive weight decay, and thus lead to significantly reduced memory utilization and computational overhead. It is observed that these optimization lead to cutting the training time by more than half, e.g., ~402 seconds and improve inference speeds through reducing redundant computations. Weighted activation functions and momentum-based gradient descent also help achieve faster convergence. Optimized DBNs achieve far higher accuracy than the basic DBNs, e.g., 98.65% as against ~79–93% with far fewer resources.

7. Empirical Comparison

The empirical comparison is a benchmark to evaluate the models' performance. Evaluating the various models enables the methodical examination and comparison of the accuracy, precision, recall, F1-score, and AUC metrics. or configurations. This will make it possible to see how well a model does on training data and how well it performs in real-world settings.

This section provides a thorough empirical comparison of an unoptimized and optimized DBN. Through an analysis of various metrics of classification, dataset-specific accuracies, and K-fold validation results, the analytical process draws attention to how optimization techniques affect a model's performance. From medical imaging to agricultural-based classification tasks, the gains in predictive accuracy, generalization, and robustness are impressive across multiple data sets. Table 1 compares the DBN and optimized DBN using the classification metrics.

Table 1. Comparison of classification metrics

Classification Metric	**DBN (Unoptimized, Above 92% Accuracy)**	**Optimized DBN (Above 98% Accuracy)**
Accuracy	93.5%	98%
Precision	0.93	0.96
Recall	0.92	0.97
F1-Score	0.925	0.965
AUC (Area Under Curve)	0.93	0.99
Sensitivity (Recall)	0.92	0.97
Specificity	0.93	0.98
Log-Loss	0.18	0.10
Balanced Accuracy	0.925	0.975
ROC Curve (AUC)	0.93	0.99

Table table 1 shows the comparison of classification metrics for two Deep Belief Networks (DBNs): an unoptimized version with over 92% accuracy and an optimized version exceeding 98% accuracy. The optimized DBN demonstrates improvements across all metrics, including accuracy (98% vs. 93.5%) and precision (0.96 vs. 0.93). Recall and F1-Score also improve significantly, indicating better performance in identifying true positives. The optimized model has a lower log loss (0.10 vs. 0.18), reflecting enhanced predictive certainty. Overall, the optimized DBN achieves higher balanced accuracy and AUC, showcasing its superior effectiveness in classification tasks.

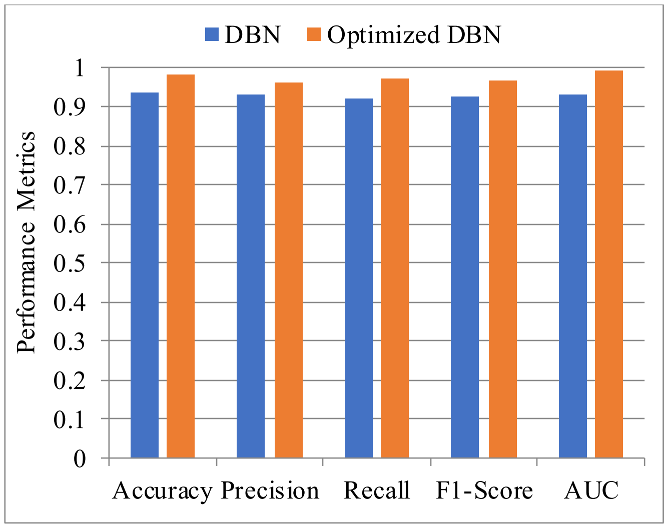

Fig. 3 Comparison of accuracy, precision, recall, F1-score and AUC

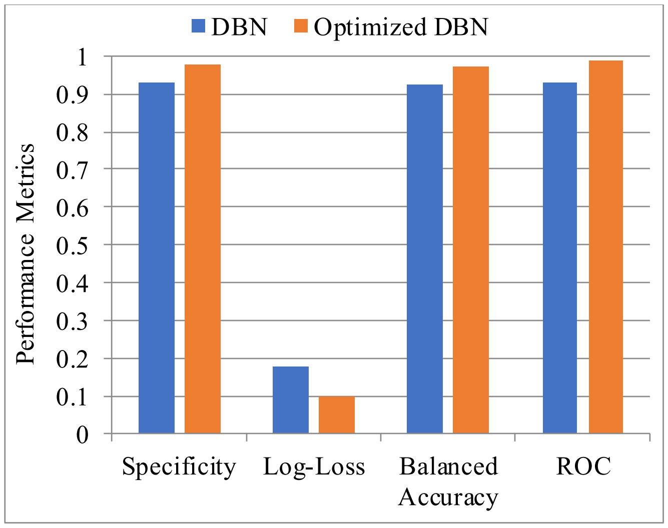

Fig. 4 Comparison of specificity, log loss, balanced accuracy and ROC

Table 2. Comparison of existing datasets

Datasets	Accuracy of DBN	Accuracy of Optimized DBN
Plant Village Datase (25)	95.5%	99%
Covid-19 Chest X-ray Dataset (26)	93.6%	98.3%
Skin Cancer MNIST: (HAM10000) Dataset (27)	92.9%	97.8%
Breast Cancer Dataset (Wisconsin) (28)	94.2%	99%
Lung Cancer Datase (29)	91.9%	97.6%
Glaucoma Fundus Imaging Datasets (30)	93.5%	98.2%
Bone Break Classification Image Dataset (31)	92.6%	97.8%

Table 2 compares the accuracy of a Deep Belief Network (DBN) and its optimized version across various datasets. The optimized DBN consistently outperforms the unoptimized model, with accuracy improvements ranging from 1.2% to 6.1%. For instance, the Plant Village Dataset sees an increase from 95.5% to 99%, while the Lung Cancer Dataset improves from 91.9% to 97.6%. All datasets show significant gains, indicating the effectiveness of optimization in enhancing classification performance. The optimized DBN achieves high accuracy across diverse medical and agricultural datasets.

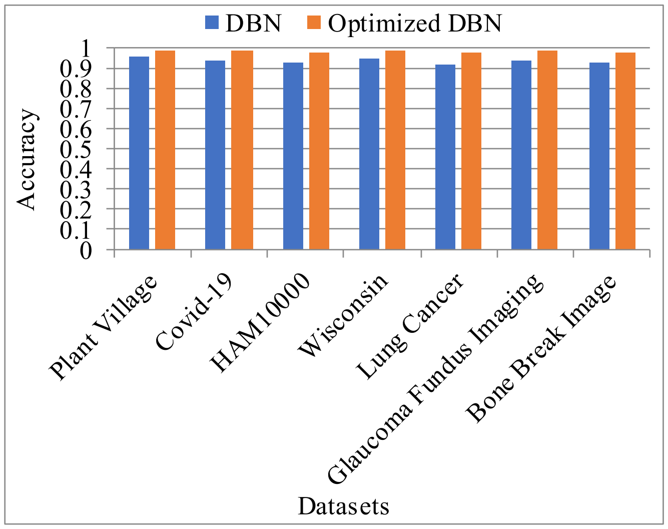

Fig. 5 Comparison of different datasets

Table 3. K-fold validation for existing dataset (K-fold value)

Datasets	DBN Accuracy (K-Fold)	Optimized DBN Accuracy (K-Fold)
Plant Village Dataset	95%	99%
COVID-19 Chest X-ray Dataset	93%	98%
Skin Cancer Dataset (HAM10000)	90%	97%
Breast Cancer Dataset (Wisconsin)	97%	99%
Lung Cancer Dataset	91%	97%
Glaucoma Fundus Imaging Datasets	93%	98%
Bone Break Classification Image Dataset	89%	95%

Table 3 presents the K-fold validation results for a Deep Belief Network (DBN) and its optimized counterpart across various datasets, using a K value of 5. The optimized DBN achieves higher accuracy in all datasets compared to the unoptimized model, with improvements ranging from 2% to

10%. For example, the accuracy for the Plant Village Dataset rises from 95% to 99%, and the Bone Break Classification Dataset improves from 89% to 95%. These results demonstrate that optimization significantly enhances the model's reliability and generalization across different data sources. Overall, the optimized DBN shows robust performance in various applications, particularly in medical and agricultural fields.

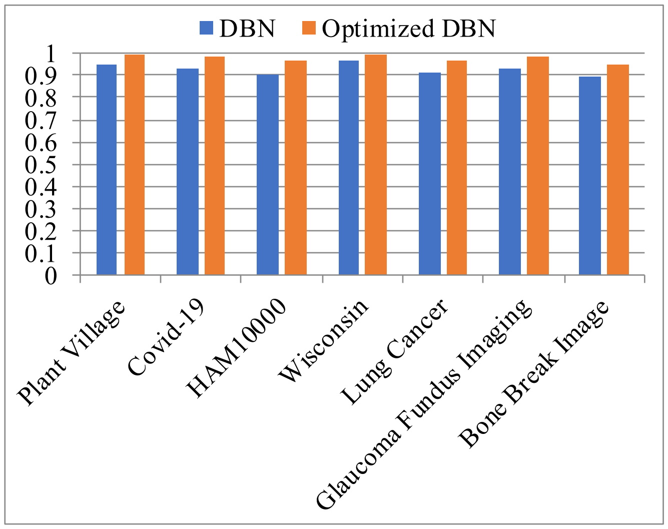

Fig. 6 Graph for K-fold validation

8. Conclusion

This comparative study illustrates the potential and shortcomings of Deep Belief Networks, which are to be compared between unoptimized and optimized architectures in a wide range of datasets. Although effective in learning hierarchical data representations, the unoptimized DBN suffers from vanishing gradients, computational inefficiency, and sensitivity to hyperparameter tuning. However, the optimized DBN utilizes state-of-the-art techniques, including adaptive learning rates, sparsity, pruning, and dynamic regularization, to overcome the stated challenges, which are important for significant improvements in terms of processing efficiency, generalization, and classification accuracy. Key insights of this study highlight the significance of optimization and fine-tuning in deep learning. Methods like supervised fine-tuning, weighted activation functions, and adaptive updates significantly improve DBN's performance in minimizing overfitting, accelerating convergence, and achieving robustness across different datasets. Empirical results show consistent superiority of the optimized DBN over the baseline version, which has more than 98% accuracy and outperforms the baseline on precision, recall, F1-score, and AUC.

Future directions for improving DBN-based architectures include the usage of advanced optimization techniques like metaheuristic algorithms, which allow for hyperparameter tuning. Transfer learning could also be further explored to expand DBNs' ability to various new domains. Lightweight variants of DBNs can also be tested for achieving real-time applications in resource-constrained devices. These new developments can more accurately shape the effectiveness and scale of DBNs, paving the way for their much wider deployment in fields of healthcare, agriculture, and many other areas.

Data Availability Statement

All the information was gathered from the authors' tools and software simulation reports. With the proper authorization, authors are striving to implement the same, utilizing real-world data.

Reference

[1] Bijaya Kumar Sethi et al., "Long Short-Term Memory-Deep Belief Network based Gene Expression Data Analysis for Prostate Cancer Detection and Classification," *IEEE Access*, vol. 12, pp. 1508-1524, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mehmet Ali Balcı et al., "A Series-Based Deep Learning Approach to Lung Nodule Image Classification," *Cancers*, vol. 15, no. 3, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Nivaashini Mathappan, Suganya Elavarasan, and Sountharrajan Sehar, "Hybrid Intelligent Intrusion Detection System for Multiple Wi-Fi Attacks in Wireless Networks using Stacked Restricted Boltzmann Machine and Deep Belief Networks," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 23, pp. 1-27, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Junhai Luo et al., "Semi-Supervised Cross-Subject Emotion Recognition Based on Stacked Denoising Autoencoder Architecture using a Fusion of Multi-Modal Physiological Signals," *Entropy*, vol. 24, no. 5, pp. 1-29, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Rini Smita Thakur et al., "Nature-Inspired DBN based Optimization Techniques for Image De-noising," *Intelligent Systems with Applications*, vol.18, pp. 1-13, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Vanlalruata Hnamte et al., "A Novel Two-Stage Deep Learning Model for Network Intrusion Detection: LSTM-AE," *IEEE Access*, vol. 11, pp. 37131 - 37148, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shijie Ren, and Feng Zhou, "Semi-Supervised Classification for PolSAR Data with Multi-Scale Evolving Weighted Graph Convolutional Network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 2911-2927, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sreenivasan Ramasamy Ramamurthy et al., "STAR-Lite: A Light-Weight Scalable Self-Taught Learning Framework for Older Adults' Activity Recognition," *Pervasive and Mobile Computing*, vol. 87, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Luis Irastorza-Valera et al., "An Agent-Based Model to Reproduce the Boolean Logic Behaviour of Neuronal Self-Organised Communities through Pulse Delay Modulation and Generation of Logic Gates," *Biomimetics*, vol. 9, no. 2, pp. 1-18, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] M.R. Ezilarasan, J. Britto Pari, and Man-Fai Leung, "Reconfigurable Architecture for Noise Cancellation in Acoustic Environment Using Single Multiply Accumulate Adaline Filter," *Electronics*, vol. 12, no. 4, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Denis Kleyko et al., "Perceptron Theory can Predict the Accuracy of Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 7, pp. 9885-9899, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Vita Santa Barletta et al., "A Kohonen SOM Architecture for Intrusion Detection on In-Vehicle Communication Networks," *Applied Sciences*, vol.10, no.15, pp. 1-27, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Meha Desai, and Manan Shah, "An Anatomization on Breast Cancer Detection and Diagnosis Employing Multi-Layer Perceptron Neural Network (MLP) and Convolutional Neural Network (CNN)," *Clinical eHealth*, vol. 4, pp. 1-11, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Boyan Li et al., "Efficient Deep Spiking Multilayer Perceptrons with Multiplication-Free Inference," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 4, pp. 7542-7554, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Siyu Lu et al., "An Improved Algorithm of Drift Compensation for Olfactory Sensors," *Applied Sciences*, vol. 12, no. 19, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Joshua O. Ighalo, Adewale George Adeniyi, and Gonçalo Marques, "Application of Artificial Neural Networks in Predicting Biomass Higher Heating Value: An Early Appraisal," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, vol. 46, no. 1, pp. 15117-15124, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] Wael Deabes, Alaa Sheta, and Malik Braik, "ECT-LSTM-RNN: An Electrical Capacitance Tomography Model-Based Long Short-Term Memory Recurrent Neural Networks for Conductive Materials," *IEEE Access*, vol. 9, pp. 76325-76339, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Roos Sophia de Freitas Dam et al., "Prediction of Fluids Volume Fraction and Barium Sulfate Scale in a Multiphase System Using Gamma Radiation and Deep Neural Network," *Applied Radiation and Isotopes*, vol. 201, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Luigi Tesio et al., "Interpreting Results from Rasch Analysis 2. Advanced Model Applications and the Data-Model Fit Assessment," *Disability and Rehabilitation*, vol. 46, no. 3, pp. 604-617, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Neha Ahlawat, and D. Franklin Vinod, "Clipped RBM and DBN Based Mechanism for Optimal Classification of Brain Cancer," *ICT with Intelligent Applications*, Singapore: Springer Nature, vol. 1, pp. 295-304, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Deliang Yu, and Huibo Zhang, "Fault Diagnosis Method for Submersible Reciprocating Pumping Unit Based on Deep Belief Network," *IEEE Access,* vol. 8, pp. 109940-109948, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] Diego Marin-Santos et al., "Automatic Detection of Crohn Disease in Wireless Capsule Endoscopic Images Using a Deep Convolutional Neural Network," *Applied Intelligence*, vol. 53, no. 10, pp. 12632-12646, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Aiguo Chen et al., "An Efficient Network Behavior Anomaly Detection Using a Hybrid DBN-LSTM Network," *computers & security*, vol. 114, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Benyamin Abdollahzadeh et al., "Puma Optimizer (PO): A Novel Metaheuristic Optimization Algorithm and its Application in Machine Learning," *Cluster Computing*, vol. 27, pp. 5235-5283, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Tairu Oluwafemi Emmanuel, PlantVillage Dataset, Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/emmarex/plantdisease

[26] Prashant Patel, Chest X-ray (Covid-19 & Pneumonia), Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/prashant268/chest-xray-covid19-pneumonia

[27] K Scott Mader, Skin Cancer MNIST: HAM10000, Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

[28] UCI Machine Learning and 1 collaborator, Breast Cancer Wisconsin (Diagnostic) Data Set, Kaggle, 2016. [Online]. Available: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

[29] Mysar Ahmad Bhat, Lung Cancer, Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer

[30] Arnav Jain, Glaucoma Fundus Imaging Datasets, Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/arnavjain1/glaucoma-datasets

[31] Parisa Karimi Darabi, Bone Break Classification Image Dataset, Kaggle, 2025. [Online]. Available: https://www.kaggle.com/datasets/pkdarabi/bone-break-classification-image-dataset